Freeform Search

Database:	US Pre-Grant Publication Full-Text Database [✓ US Patents Full-Text Database] US OCR Full-Text Database EPO Abstracts Database JPO Abstracts Database Derwent World Patents Index IBM Technical Disclosure Bulletins
Term:	L11 AND 16
Display:	50 **Documents in Display Format:** REV **Starting with Number** 1
Generate:	Hit List [✓ Hit Count] Side by Side Image

Search Clear Interrupt

Search History

DATE: Wednesday, September 14, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L12	L11 AND 16	0	L12
L11	L10 OR 19	6	L11
L10	L8 and instrumentation.ti.	3	L10
L9	L8 and profiler.ti.	3	L9
L8	(PROFILE OR profiler OR instrumentation) AND (ingberg.XP. Or ingberg.xa.)	48	L8
L7	(PROFILE OR profiler) AND (ingberg.XP. Or ingberg.xa.)	35	L7
L6	L5 ANd 717/130-133.ccls.	16	L6
L5	L4 ANd ((user ADJ defined) OR (scalar))	827	L5
L4	L3 AND range	8350	L4
L3	L2 AND software	10513	L3
L2	profiler OR instrumentation	46908	L2
L1	6732357.pn. OR 6728955.pn.	2	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs
Generate OACS

Search Results - Record(s) 1 through 16 of 16 returned.

☐ 1. Document ID: US 6941545 B1

```
L6: Entry 1 of 16                     File: USPT                    Sep 6, 2005

US-PAT-NO: 6941545
DOCUMENT-IDENTIFIER: US 6941545 B1

TITLE: Profiling of computer programs executing in virtual memory systems

DATE-ISSUED: September 6, 2005

INVENTOR-INFORMATION:
NAME                            CITY                 STATE   ZIP CODE    COUNTRY
Reese; David L.                 Westborough          MA
Yates, Jr.; John S.             Needham              MA
Hohensee; Paul H.               Nashua               NH
Van Dyke; Korbin S.             Sunol                CA
Ramesh; T. R.                   Newark               CA
Thusoo; Shalesh                 Milpitas             CA
Saund; Gurjeet Singh            Mountain View        CA
Patkar; Niteen Aravind          Sunnyvale            CA

US-CL-CURRENT: 717/130; 711/206, 711/213, 712/207, 712/227, 717/127, 717/128,
717/131, 717/136, 717/140

ABSTRACT:
```

A computer. An instruction pipeline and memory access unit execute instructions in a logical address space of a memory of the computer. An address translation circuit translates address references generated by the program from the program's logical address space to the computer's physical address space. Profile circuitry is cooperatively interconnected with the instruction pipeline and configured to detect, without compiler assistance for execution profiling, occurrence of profilable events occurring in the instruction pipeline, and is cooperatively interconnected with the memory access unit to record profile information describing physical memory addresses referenced during an execution interval of the program.

```
81 Claims, 37 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 32
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw D

☐ 2. Document ID: US 6895578 B1

L6: Entry 2 of 16 File: USPT May 17, 2005

US-PAT-NO: 6895578
DOCUMENT-IDENTIFIER: US 6895578 B1

TITLE: Modularizing a computer program for testing and debugging

DATE-ISSUED: May 17, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kolawa; Adam K.	Bradbury	CA		
Byers; Chad E.	San Luis Obispo	CA		

US-CL-CURRENT: 717/130; 717/134, 717/143

ABSTRACT:

A system and method for facilitating and simplifying testing and debugging of computer programs. is described A computer program is broken down to smaller components, such as, classes, functions, or objects, and then those smaller components are tested individually. Accordingly, specific aspects of the computer program can be effectively tested. The user can automatically perform a range of tests on a class or method when the class or method is compiled without integrating the class or method into a larger project.

37 Claims, 30 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 27

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw D

☐ 3. Document ID: US 6865429 B1

L6: Entry 3 of 16 File: USPT Mar 8, 2005

US-PAT-NO: 6865429
DOCUMENT-IDENTIFIER: US 6865429 B1

TITLE: Real-time control system development tool

DATE-ISSUED: March 8, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Schneider; Stanley A.	Sunnyvale	CA		
Chen; Vincent W.	San Jose	CA		
Pardo-Castellote; Gerardo	Palo Alto	CA		
Wang; Howard H.	Sunnyvale	CA		

Joshi; Rajive	Sunnyvale	CA

US-CL-CURRENT: 700/86; 700/87, 717/104, 717/107, 717/120, 717/132, 717/153, 717/155

ABSTRACT:

A composite object group (COG) data structure embodied in a computer-readable medium for building a control system that has both a clock cycle and event processing is provided. An interface for passing information to and from the COG data structure is provided. One or more data flow objects are provided in the COG to accept input data and to produce output data on the clock cycle. The data flow object is connected to the interface and provides sampled-data processing for the control system. One or more state machine objects are provided in the COG; each includes a plurality of states and a plurality of transitions between the states that are each triggered by an event. The state machine object provides event-driven processing for the control system, whereby the COG data structure provides both sampled-data and event-driven processing for the control system.

20 Claims, 34 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 24

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw D

☐ 4. Document ID: US 6826748 B1

L6: Entry 4 of 16 File: USPT Nov 30, 2004

US-PAT-NO: 6826748
DOCUMENT-IDENTIFIER: US 6826748 B1

TITLE: Profiling program execution into registers of a computer

DATE-ISSUED: November 30, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hohensee; Paul H.	Nashua	NH		
Reese; David L.	Westborough	MA		
Yates, Jr.; John S.	Needham	MA		
Van Dyke; Korbin S.	Sunol	CA		
Ramesh; T. R.	Newark	CA		
Thusoo; Shalesh	Milpitas	CA		
Saund; Gurjeet Singh	Mountain View	CA		
Patkar; Niteen Aravind	Sunnyvale	CA		

US-CL-CURRENT: 717/130

ABSTRACT:

A method and computer for performance of the method. While executing a program on a computer, the computer uses registers of a general register file for storage of

instruction results. Profile information describing the profileable events is recorded into the general register file as the profileable events occur, without first capturing the information into a main memory of the computer.

27 Claims, 37 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 32

☑ 5. Document ID: US 6671830 B2

L6: Entry 5 of 16 File: USPT Dec 30, 2003

US-PAT-NO: 6671830
DOCUMENT-IDENTIFIER: US 6671830 B2

TITLE: Method and apparatus for analyzing performance of data processing system

DATE-ISSUED: December 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kaler; Christopher G.	Redmond	WA		
Lowell; Martyn S.	Seattle	WA		
Wahbe; Robert S.	Seattle	WA		
Ferguson; William J.	Bellevue	WA		
Sharp; Oliver J.	New York	NY		

US-CL-CURRENT: 714/39; 714/47, 717/127, 717/130, 719/318

ABSTRACT:

A method and apparatus for analyzing the performance of a data processing system, particularly a distributed data processing system, provide a system user with tools for analyzing an application running thereon. Information about the flow and performance of the application can be specified, captured, and analyzed, without modifying it or degrading its performance or data security characteristics, even if it is distributed across multiple machines. The user interface permits the system user to filter the performance information, to set triggers which the performance analyzer is able to reduce and/or combine, to observe multiple time-synchronized displays of performance data either in real time or post mortem, and to play and re-play the operation of an automatically generated application model. The invention is implemented in part by providing suitable Application Program Interfaces (APIs) in the operating system of the data processing system.

17 Claims, 38 Drawing figures
Exemplary Claim Number: 15
Number of Drawing Sheets: 33

☑ 6. Document ID: US 6493868 B1

L6: Entry 6 of 16 File: USPT Dec 10, 2002

US-PAT-NO: 6493868
DOCUMENT-IDENTIFIER: US 6493868 B1

TITLE: Integrated development tool

DATE-ISSUED: December 10, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
DaSilva; Greg N.	Rexdale			CA
Gingrich; Paul	Toronto			CA
Pandey; Raju	Toronto			CA

US-CL-CURRENT: 717/105; 345/1.3, 345/540, 717/108, 717/111, 717/129, 717/130

ABSTRACT:

An integrated code development tool, comprising of an editor, a project management and build system, a debugger, a profiler, and a graphical data visualization system. The editor is operable to provide a source code view which is simultaneously capable of integrating with said debugger to provide for stepping through code and setting breakpoints, and integrating with the output of said build system to display source code interleaved with corresponding assembler code created by said build system.

19 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | [illegible] | [illegible] | Claims | KWIC | Draw De

☑ 7. Document ID: US 6467052 B1

L6: Entry 7 of 16 File: USPT Oct 15, 2002

US-PAT-NO: 6467052
DOCUMENT-IDENTIFIER: US 6467052 B1
**** See image for Certificate of Correction ****

TITLE: Method and apparatus for analyzing performance of data processing system

DATE-ISSUED: October 15, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kaler; Christopher G.	Redmond	WA		

Lovell; Martyn S.	Seattle	WA
Wahbe; Robert S.	Seattle	WA
Ferguson; William J.	Bellevue	WA
Sharp; Oliver J.	New York	NY

US-CL-CURRENT: 714/39; 714/47, 717/127, 717/130, 719/318

ABSTRACT:

A method and apparatus for analyzing the performance of a data processing system, particularly a distributed data processing system, provide a system user with tools for analyzing an application running thereon. Information about the flow and performance of the application can be specified, captured, and analyzed, without modifying it or degrading its performance or data security characteristics, even if it is distributed across multiple machines. The user interface permits the system user to filter the performance information, to set triggers which the performance analyzer is able to reduce and/or combine, to observe multiple time-synchronized displays of performance data either in real time or post mortem, and to play and re-play the operation of an automatically generated application model. The invention is implemented in part by providing suitable Application Program Interfaces (APIs) in the operating system of the data processing system.

31 Claims, 38 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 33

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw. D

☑ 8. Document ID: US 6149318 A

L6: Entry 8 of 16 File: USPT Nov 21, 2000

US-PAT-NO: 6149318
DOCUMENT-IDENTIFIER: US 6149318 A

TITLE: Link-time and run-time error detection, and program instrumentation

DATE-ISSUED: November 21, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chase; David R.	Belmont	MA		
Kendall; Samuel C.	Cambridge	MA	02139	
Mitchell; Mark Patrick	Atherton	CA		

US-CL-CURRENT: 717/131; 717/116

ABSTRACT:

A programming language processor performs link-time and run-time error checking of a program written in C, C++, or a combination of both. The link-time error checking diagnoses violations of the C++ One Definition Rule, and its equivalent in C. As

the program runs, the run-time error checking examines accesses to computer memory to determine that the addresses accessed contain values of che type expected by the program. To add instrumentation to a C or C++ program, pre-expressions, post-expressions and clone-expressions are used to annotate an abstract syntax tree, the annotated tree is then canonicalized into a more traditional syntax tree before a back-end generates code for the program.

6 Claims, 35 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 27

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw D

☑ 9. Document ID: US 6085029 A

L6: Entry 9 of 16 File: USPT Jul 4, 2000

US-PAT-NO: 6085029
DOCUMENT-IDENTIFIER: US 6085029 A

TITLE: Method using a computer for automatically instrumenting a computer program for dynamic debugging

DATE-ISSUED: July 4, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kolawa; Adam K.	Sierra Madre	CA		
Salvador; Roman	Barcelona			ES
Hicken; Wendell T.	Whittier	CA		
Strickland; Bryan R.	Los Angeles	CA		

US-CL-CURRENT: 714/38; 717/130, 717/141, 717/144

ABSTRACT:

A method for automatically instrumenting a computer program for dynamic debugging. Such a computer program comprising source code written in a programming language for executing instructions on the computer. The source code is provided as a sequence of statements in a storage device to the computer. Each of the statements are separated into tokens representing either an operator or at least one operand. A parse tree is built according to a set of rules using the set of tokens. The parse tree is instrumented to create an instrumented parse tree for indicating that an error condition occurred in the computer program during execution. Object code is generated from the instrumented parse tree and stored in a secondary storage device for later execution using an error-checking engine that indicates error conditions present in the computer program.

50 Claims, 41 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 32

Search Forms
Search Results

☑ 10. Document ID: US 6079032 A

L6: Entry 10 of 16 File: USPT Jun 20, 2000

US-PAT-NO: 6079032
DOCUMENT-IDENTIFIER: US 6079032 A

TITLE: Performance analysis of computer systems

DATE-ISSUED: June 20, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Peri; Ramesh V.	Allentown	PA		

US-CL-CURRENT: 714/38; 717/129, 717/131

ABSTRACT:

A method of analyzing performance of a program executing in a computer system. A user provides a set of user defined region of the program. Thus, a user has the flexibility to choose the regions of program code profiled. The performance of user defined regions of the program is measured by a set of run-time metrics. Each user defined region is associated with a range break point. Run-time metrics measuring the performance of user defined regions of the program are updated, during execution, whenever a range break point is set. The handling of range break points may be implemented, for example, by specialized hardware and software. This method may be less intrusive than instrumentation based profiling but more accurate than sampling based profiling.

14 Claims, 10 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 11. Document ID: US 6077311 A

L6: Entry 11 of 16 File: USPT Jun 20, 2000

US-PAT-NO: 6077311
DOCUMENT-IDENTIFIER: US 6077311 A

TITLE: Method and apparatus for extraction of program region

DATE-ISSUED: June 20, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY

Lichtenstein; Walter D.	Belmont	MA
Dahl; Rune	San Francisco	CA
Towle; Ross	San Francisco	CA

US-CL-CURRENT: 717/128; 717/130, 717/133, 717/146

ABSTRACT:

A method and apparatus for marking a region of source code within a program unit and extracting an executable version of this marked region of code. The executable version has a initialized program state equivalent to that of the original source code when the original source code entered the region. The method and apparatus remove as much source code as possible from the original source code while retaining enough code to enable the extracted region execute (replay) in a manner identical to the execution of the program region in the context of the original system.

14 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw D

☐ 12. Document ID: US 6070009 A

L6: Entry 12 of 16 File: USPT May 30, 2000

US-PAT-NO: 6070009
DOCUMENT-IDENTIFIER: US 6070009 A

TITLE: Method for estimating execution rates of program execution paths

DATE-ISSUED: May 30, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dean; Jeffrey	Menlo Park	CA		
Eustace; Robert A.	Redwood City	CA		
Hicks; James E.	Newton	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 717/130; 717/132, 717/154, 717/159

ABSTRACT:

A method is provided for estimating execution rates of program executions paths. The method samples path-identifying state information of selected instructions while executing the program in a processor. A control flow graph of the program is supplied, the control flow graph includes a plurality of path segments. The control flow graph is analyzed using the path-identifying state information to identify a set of path segments that are consistent with the sampled state information. The

set of paths segments can be counted to determine their relative execution frequencies.

22 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 20

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw De

---

☐ 13. Document ID: US 5903758 A

L6: Entry 13 of 16 File: USPT May 11, 1999

US-PAT-NO: 5903758
DOCUMENT-IDENTIFIER: US 5903758 A

TITLE: Method and apparatus for auditing dynamically linked procedure calls

DATE-ISSUED: May 11, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Walker; Michael S.	Mountain View	CA		

US-CL-CURRENT: 717/130; 714/35, 717/133, 719/328, 719/331

ABSTRACT:

The present invention is a method and apparatus for providing instrumentation of procedure calls in dynamically linked environments. More specifically, an embodiment of the present invention includes an API that allows a user to define procedures that are called during specific times during the execution of a runtime linker. By defining procedures in accordance with this API, the user can select procedures within a user program for auditing. The API also allows the user to define an entry procedure that will be called immediately before each audited procedure and an exit procedure that will be called immediately after each audited procedure. The runtime linker uses the procedures defined by the user to select procedures within the program for auditing. The runtime linker then arranges for the entry procedure to be called before, and the exit procedure to be called after, each audited procedure.

21 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 5

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw De

---

☐ 14. Document ID: US 5732277 A

L6: Entry 14 of 16 File: USPT Mar 24, 1998

US-PAT-NO: 5732277
DOCUMENT-IDENTIFIER: US 5732277 A
**** See image for Certificate of Correction ****

TITLE: Graphical system for modelling a process and associated method

DATE-ISSUED: March 24, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kodosky; Jeffrey L.	Austin	TX		
Truchard; James J.	Austin	TX		
MacCrisken; John E.	Palo Alto	CA		

US-CL-CURRENT: 717/125; 715/759, 715/835, 717/132, 717/134

ABSTRACT:

A method for programming a computer to execute a procedure is based on a graphical interface which utilizes data flow diagrams to represent the procedure. The method stores a plurality of executable functions, scheduling functions, and data types. A data flow diagram is assembled in response to the user input utilizing icons which correspond to the respective executable functions, scheduling functions, and data types which are interconnected by arcs on the screen. A panel, representative of an instrument fron panel having input and output formats is likewise assembled for the data flow diagram. An executable program is generated in response to the data flow diagram and the panel utilizing the executable functions, scheduling functions, and data types stored in the memory. Furthermore, the executable functions may include user defined functions that have been generated using the method for programming. In this manner, a hierarchy of procedures is implemented, each represented by a data flow diagram.

139 Claims, 125 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 64

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw De

---

☐ 15. Document ID: US 5450586 A

L6: Entry 15 of 16 File: USPT Sep 12, 1995

US-PAT-NO: 5450586
DOCUMENT-IDENTIFIER: US 5450586 A

TITLE: System for analyzing and debugging embedded software through dynamic and interactive use of code markers

DATE-ISSUED: September 12, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY

Kuzara; Eric J.	Colorado Springs	CO
Blasciak; Andrew J.	Colorado Springs	CO
Parets; Greg S.	Loveland	CO

US-CL-CURRENT: 717/124; 714/1, 717/133

ABSTRACT:

A system for inserting code markers for observing indications (external to the microprocessor upon which the software operates) of the occurrence of an event in the execution of the software. Additional instructions or markers are added to the software to be debugged to produce simple, encoded, memory references to otherwise unused memory or I/O locations that will always be visible to a logic analyzer as bus cycles. Although the code markers cause a minimal intrusion in the underlying software, they make tracing events by a conventional logic analyzer much simpler and allow for performance evaluations in manners not heretofore possible. The inserted code markers provide a method of dynamically extracting information from a running host or real-time "black box" embedded system under test using simple low intrusion print statements, encoded I/O writes on procedure entries and exits, and/or an interface to service calls and the like which writes out the passed parameters. The code markers are inserted at compile time or interactively during the debug session to make visible critical points in the code execution, such as function calls, task creation and semaphore operations, so as to speed isolation of problems at test points during debugging. Performance analysis and event analysis use the code markers to cut through the ambiguities of microprocessor prefetch and cache operations. Because of these features, the invention is particularly advantageous for use by software design teams developing complex embedded host or real-time operating systems using multi-task operating systems and/or object oriented systems.

16 Claims, 11 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 10

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw D

☐ 16. Document ID: US 5274811 A

L6: Entry 16 of 16 File: USPT Dec 28, 1993

US-PAT-NO: 5274811
DOCUMENT-IDENTIFIER: US 5274811 A

TITLE: Method for quickly acquiring and using very long traces of mixed system and user memory references

DATE-ISSUED: December 28, 1993

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Borg; Anita	San Mateo County	CA		
Wall; David W.	San Mateo County	CA		

US-CL-CURRENT: 717/128; 717/133

ABSTRACT:

The present invention utilizes link time code modification to instrument the code which is to be executed, typically comprising plurality of kernel operations and user programs. When the code is instrumented, wherever a data memory reference appears, the linker inserts a very short stylized subroutine call to a routine that logs the reference in a large, trace buffer. The same call is inserted at the beginning of each basic block to record instruction references. When the trace buffer fills up with recorded memory references, the contents of the buffer are processed, either by dumping the contents to an output device emptying the trace buffer, or a cache simulation routine is run to analyze the data. The results of the analysis are stored rather than storing the entire results of the tracing program.

16 Claims, 19 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

Terms	Documents
L5 ANd (717/130 |717/131 |717/132 |717/133).ccls.	16